National Textile University

Department of Computer Science

Subject:

Operating System

Submitted to:

Sir Nasir Mehmood

Submitted by:

Eman Babar

Reg. number:

23-NTU-CS-FL-1148

Semester:

5th- A

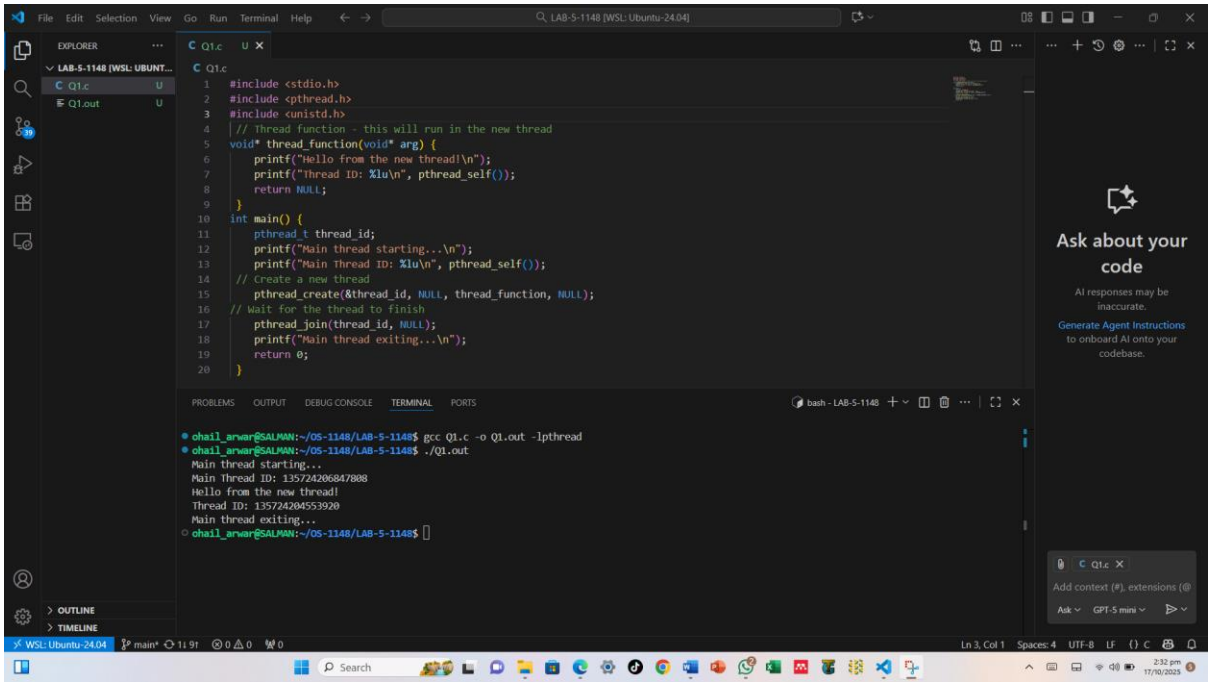
LAB-05: Introduction to Threads

Program 1: Creating a Simple Thread

Code:

```
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
 // Thread function - this will run in the new thread
void* thread_function(void* arg) {
    printf("Hello from the new thread!\n");
    printf("Thread ID: %lu\n", pthread_self());
    return NULL;
 }
int main() {
    pthread_t thread_id;
    printf("Main thread starting...\n");
    printf("Main Thread ID: %lu\n", pthread_self());
 // Create a new thread
    pthread_create(&thread_id, NULL, thread_function, NULL);
// Wait for the thread to finish
    pthread_join(thread_id, NULL);
    printf("Main thread exiting...\n");
    return 0;
 }
```

Output:



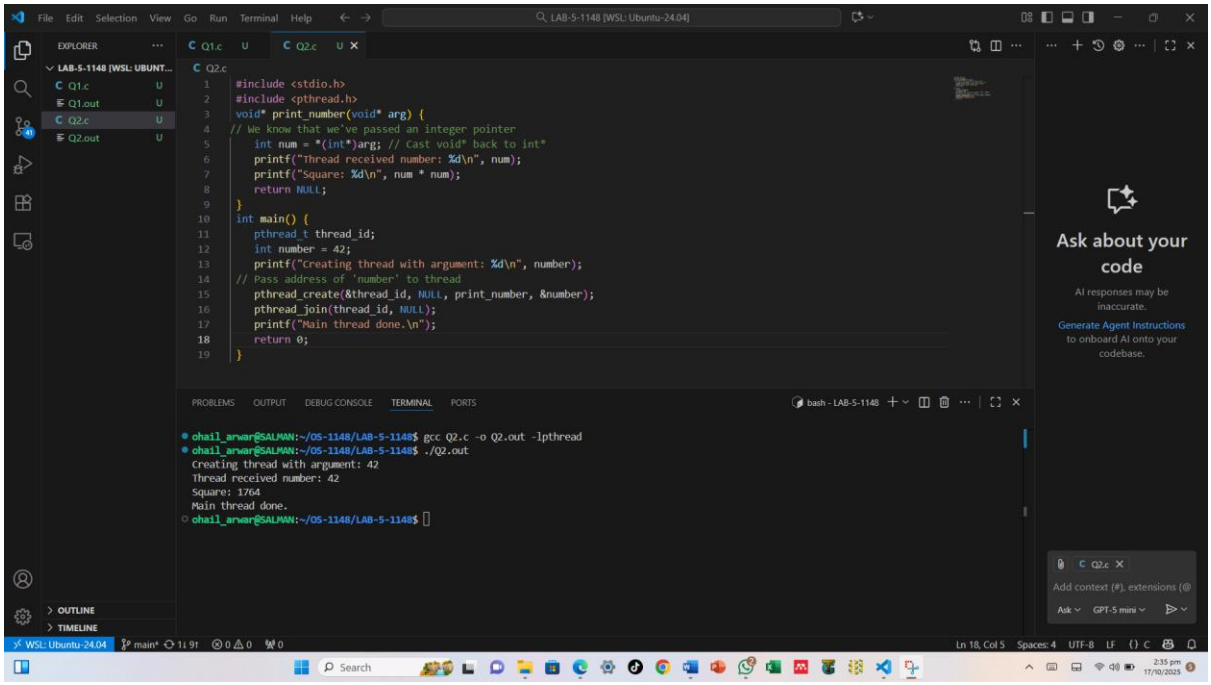
Program 2: Passing Arguments to Threads

Code:

```
#include <stdio.h>
 #include <pthread.h>
 void* print_number(void* arg) {
```

```
// We know that we've passed an integer pointer
    int num = *(int*)arg; // Cast void* back to int*
    printf("Thread received number: %d\n", num);
    printf("Square: %d\n", num * num);
    return NULL;
 }
 int main() {
    pthread_t thread_id;
    int number = 42;
    printf("Creating thread with argument: %d\n", number);
 // Pass address of 'number' to thread
    pthread_create(&thread_id, NULL, print_number, &number);
    pthread_join(thread_id, NULL);
    printf("Main thread done.\n");
    return 0;
 }
```

Output:



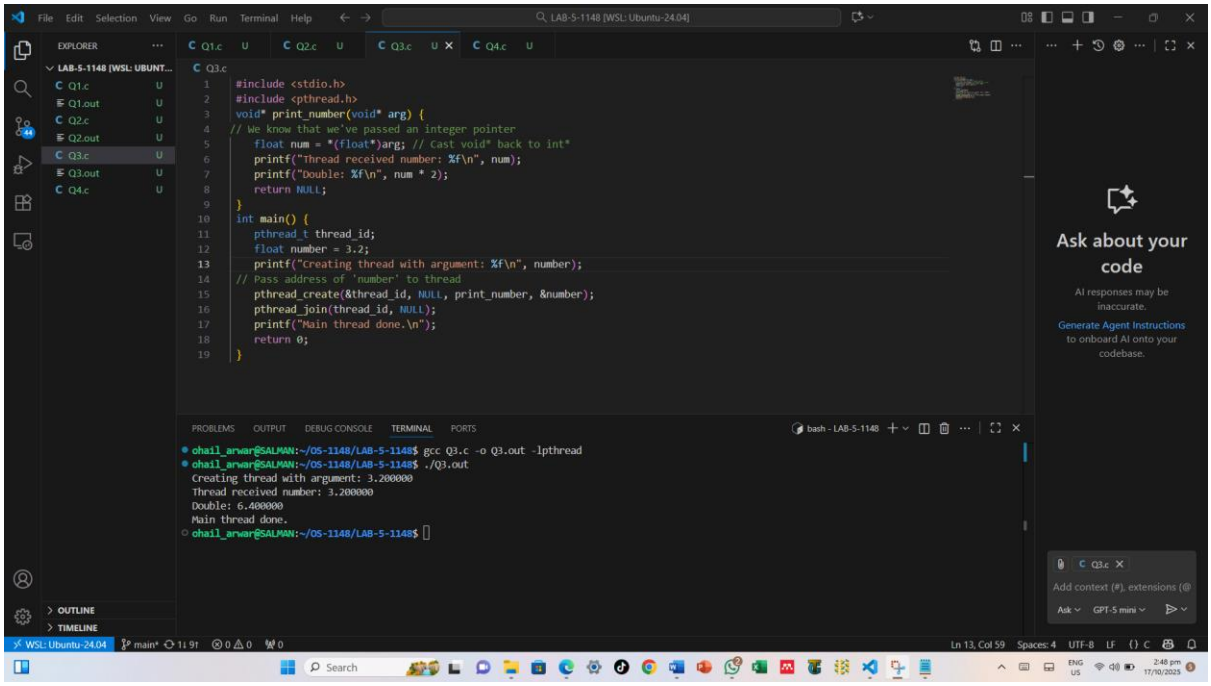
Program 3: Doubling the CGPA

Code:

```
#include <stdio.h>
 #include <pthread.h>
 void* print_number(void* arg) {
// We know that we've passed an integer pointer
    float num = *(float*)arg; // Cast void* back to int*
    printf("Thread received number: %f\n", num);
    printf("Double: %f\n", num * 2);
    return NULL;
 }
 int main() {
    pthread_t thread_id;
    float number = 3.2;
    printf("Creating thread with argument: %f\n", number);
 // Pass address of 'number' to thread
```

```
    pthread_create(&thread_id, NULL, print_number, &number);
    pthread_join(thread_id, NULL);
    printf("Main thread done.\n");
    return 0;
 }
```

Output:

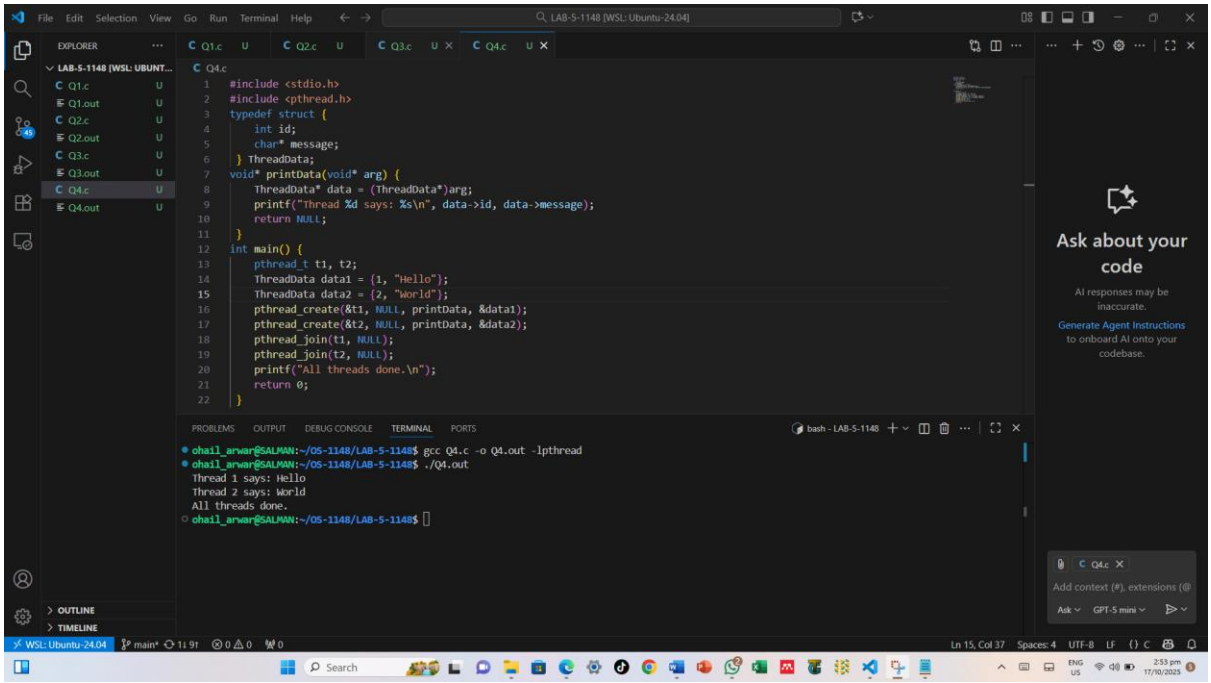


Program 4: Passing Multiple Data

Code:

```
#include <stdio.h>
#include <pthread.h>
typedef struct {
    int id;
    char* message;
 } ThreadData;
void* printData(void* arg) {
    ThreadData* data = (ThreadData*)arg;
    printf("Thread %d says: %s\n", data->id, data->message);
    return NULL;
 }
int main() {
    pthread_t t1, t2;
    ThreadData data1 = {1, "Hello"};
    ThreadData data2 = {2, "World"};
    pthread_create(&t1, NULL, printData, &data1);
    pthread_create(&t2, NULL, printData, &data2);
    pthread_join(t1, NULL);
    pthread_join(t2, NULL);
    printf("All threads done.\n");
    return 0;
 }
```

Output:

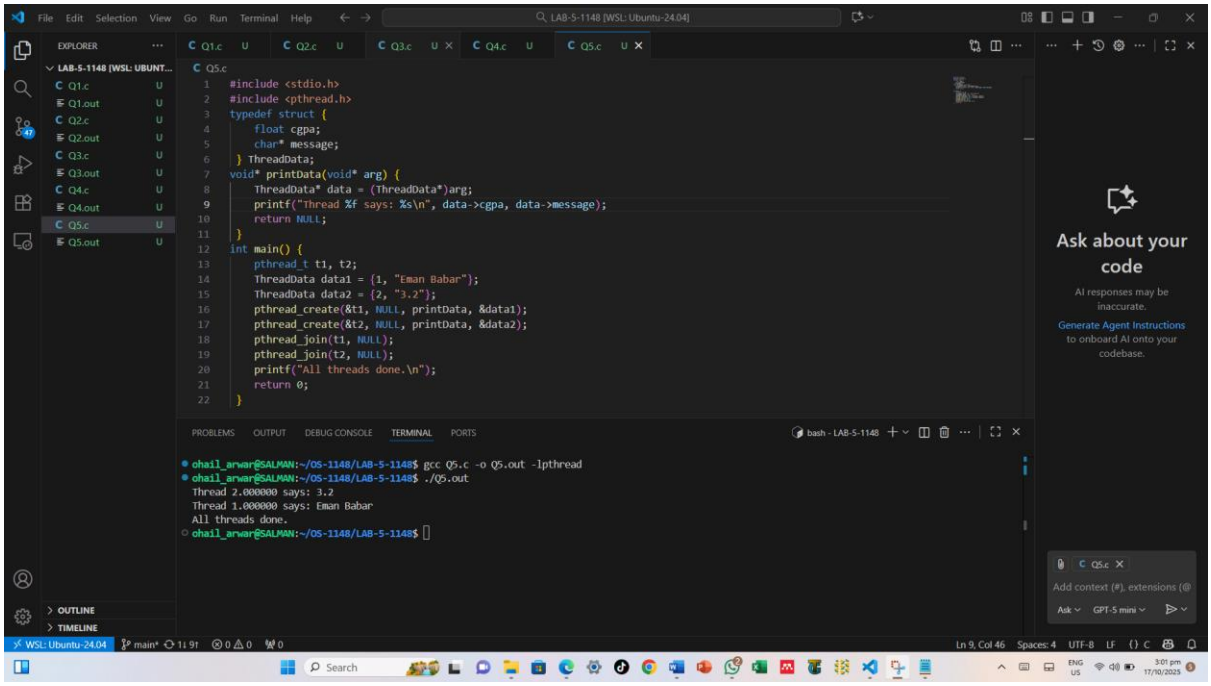


Program 5: Showing Name and CGPA

Code:

```
#include <stdio.h>
#include <pthread.h>
typedef struct {
    float cgpa;
    char* message;
 } ThreadData;
void* printData(void* arg) {
    ThreadData* data = (ThreadData*)arg;
    printf("Thread %f says: %s\n", data->cgpa, data->message);
    return NULL;
 }
int main() {
    pthread_t t1, t2;
    ThreadData data1 = {1, "Eman Babar"};
    ThreadData data2 = {2, "3.2"};
    pthread_create(&t1, NULL, printData, &data1);
    pthread_create(&t2, NULL, printData, &data2);
    pthread_join(t1, NULL);
    pthread_join(t2, NULL);
    printf("All threads done.\n");
    return 0;
 }
```

Output:

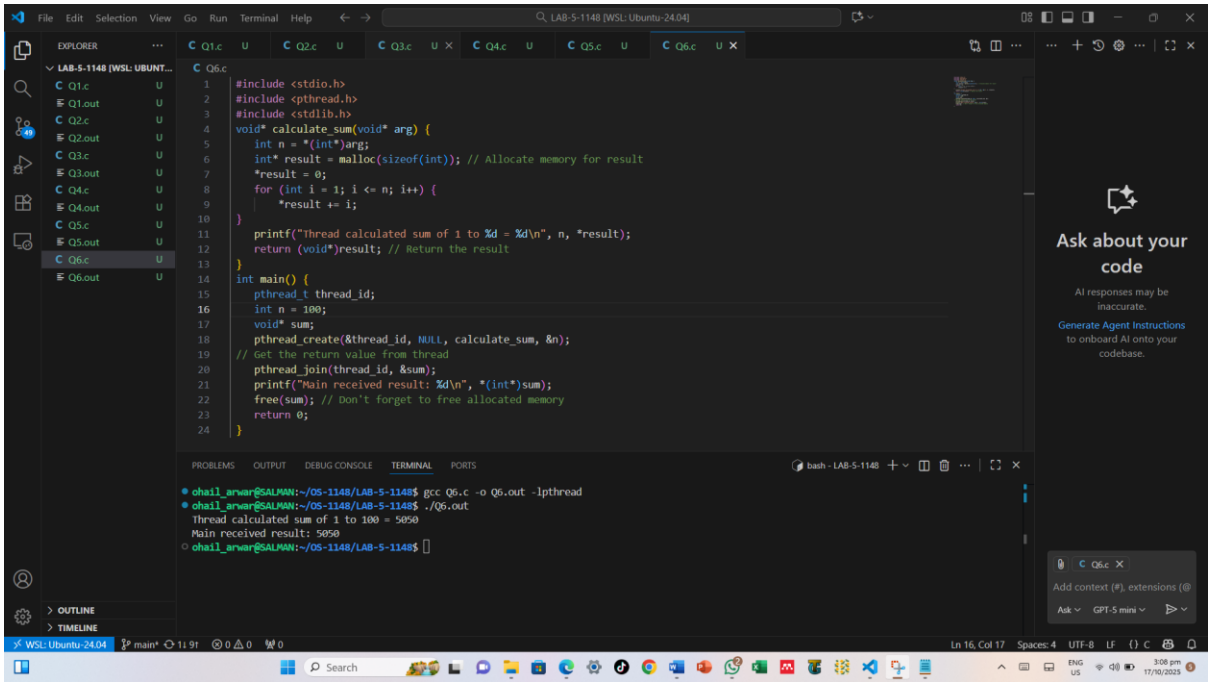


Program 6: Thread Return Values

Code:

```
#include <stdio.h>
 #include <pthread.h>
 #include <stdlib.h>
 void* calculate_sum(void* arg) {
    int n = *(int*)arg;
    int* result = malloc(sizeof(int)); // Allocate memory for result
    *result = 0;
    for (int i = 1; i <= n; i++) {
        *result += i;
 }
    printf("Thread calculated sum of 1 to %d = %d\n", n, *result);
    return (void*)result; // Return the result
 }
 int main() {
    pthread_t thread_id;
    int n = 100;
    void* sum;
    pthread_create(&thread_id, NULL, calculate_sum, &n);
 // Get the return value from thread
    pthread_join(thread_id, &sum);
    printf("Main received result: %d\n", *(int*)sum);
    free(sum); // Don't forget to free allocated memory
    return 0;
 }
```

Output:

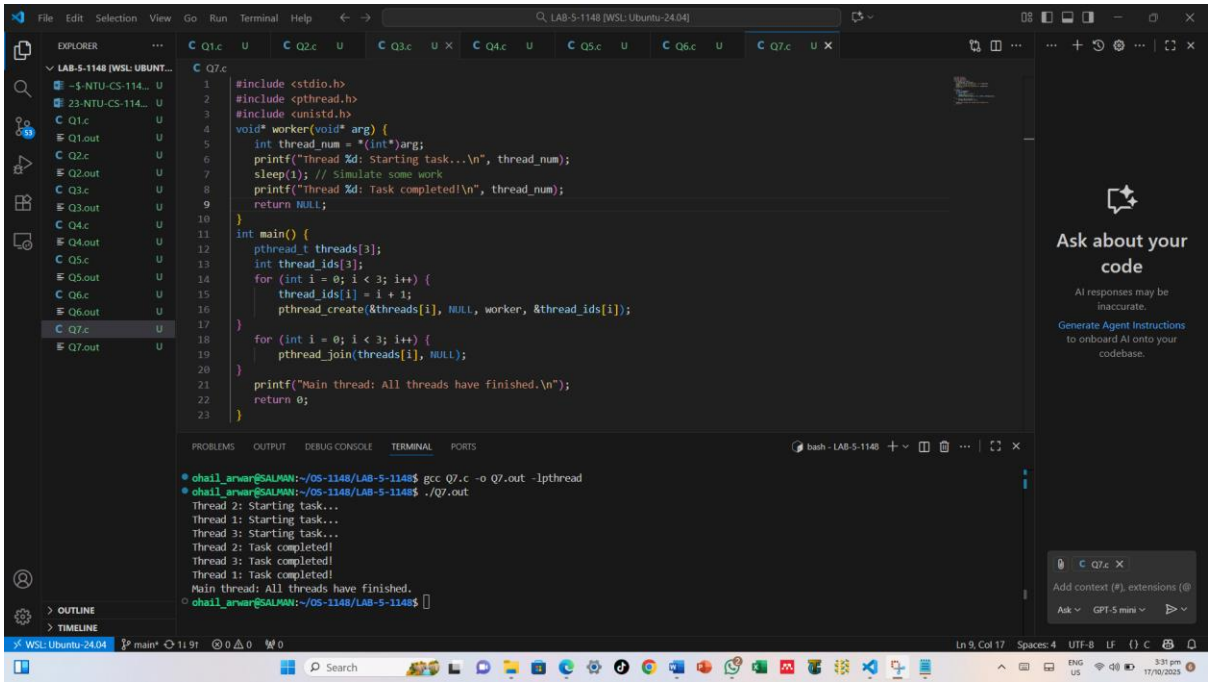


Program 7: Thread Return Values

Code:

```
#include <stdio.h>
 #include <pthread.h>
 #include <unistd.h>
 void* worker(void* arg) {
    int thread_num = *(int*)arg;
    printf("Thread %d: Starting task...\n", thread_num);
    sleep(1); // Simulate some work
    printf("Thread %d: Task completed!\n", thread_num);
    return NULL;
 }
 int main() {
    pthread_t threads[3];
    int thread_ids[3];
    for (int i = 0; i < 3; i++) {
        thread_ids[i] = i + 1;
        pthread_create(&threads[i], NULL, worker, &thread_ids[i]);
 }
    for (int i = 0; i < 3; i++) {
        pthread_join(threads[i], NULL);
 }
    printf("Main thread: All threads have finished.\n");
    return 0;
 }
```

Output:

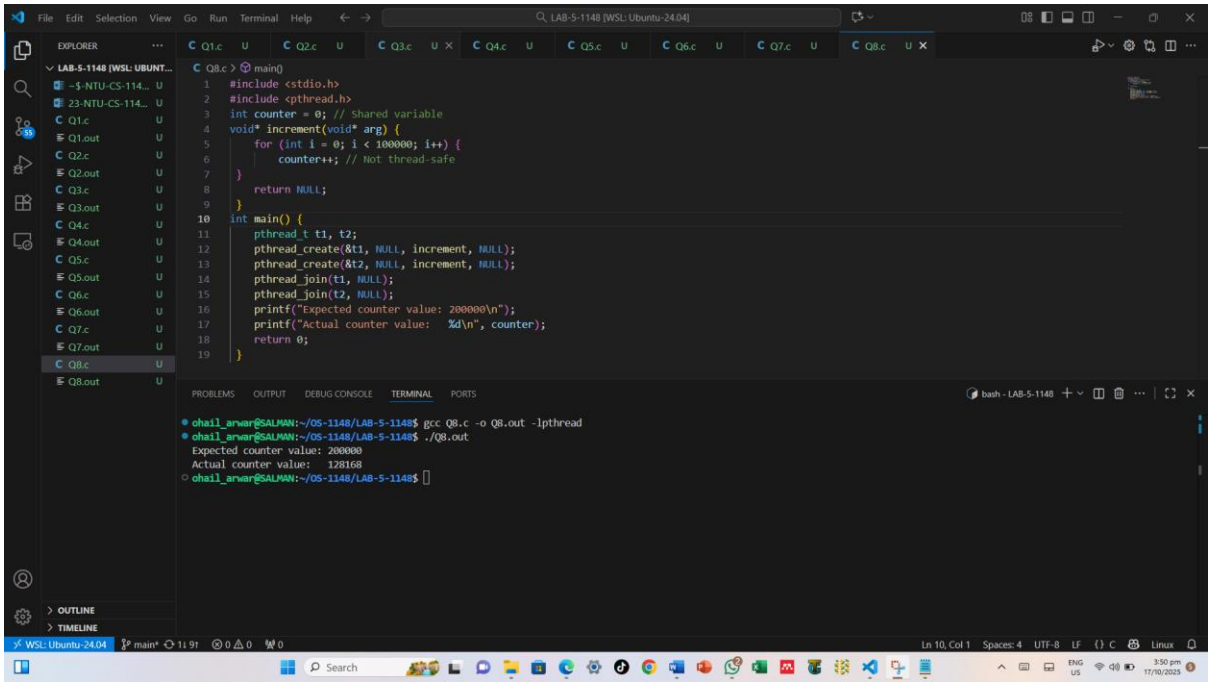


Program 8: Demonstrating a Race Condition

Code:

```
#include <stdio.h>
 #include <pthread.h>
 int counter = 0; // Shared variable
 void* increment(void* arg) {
    for (int i = 0; i < 100000; i++) {
        counter++; // Not thread-safe
 }
    return NULL;
 }
 int main() {
    pthread_t t1, t2;
    pthread_create(&t1, NULL, increment, NULL);
    pthread_create(&t2, NULL, increment, NULL);
    pthread_join(t1, NULL);
    pthread_join(t2, NULL);
    printf("Expected counter value: 200000\n");
    printf("Actual counter value:   %d\n", counter);
    return 0;
 }
```

Output:


Q8.c > main()
#include <stdio.h>
#include <pthread.h>
int counter = 0; // Shared variable
void* increment(void* arg) {
for (int i = 0; i < 100000; i++) {
counter++; // Not thread-safe
}
return NULL;
}
int main() {
pthread_t t1, t2;
pthread_create(&t1, NULL, increment, NULL);
pthread_create(&t2, NULL, increment, NULL);
pthread_join(t1, NULL);
pthread_join(t2, NULL);
printf("Expected counter value: 200000\n");
printf("Actual counter value: %d\n", counter);
return 0;
}
ohail_arwar@SALMAN:~/OS-1148/LAB-5-1148$ gcc Q8.c -o Q8.out -lpthread
ohail_arwar@SALMAN:~/OS-1148/LAB-5-1148$ ./Q8.out
Expected counter value: 200000
Actual counter value: 128168
ohail_arwar@SALMAN:~/OS-1148/LAB-5-1148$